An Adaptive Scheduling Protocol for Multi-scale Sensor Network Architecture*

Santashil PalChaudhuri[1] and David B. Johnson[2]

[1] Aruba Networks, Sunnyvale, CA
[2] Department of Computer Science, Rice University, Houston, TX

Abstract. In self-organizing networks of battery-powered wireless sensors that can sense, process, and communicate, energy is the most crucial and scarce resource. However, since sensor network applications generally exhibit specific limited behaviors, there is both a need and an opportunity for adapting the network architecture to match the application in order to optimize resource utilization. Many applications–such as large-scale collaborative sensing, distributed signal processing, and distributed data assimilation–require sensor data to be available at multiple resolutions, or allow fidelity to be traded-off for energy efficiency. We believe that cross-layering and application-specific adaptability are the primary design principles needed to build sensor networking protocols. In previous work, we proposed an adaptive cross-layered self-organizing hierarchical data service under COMPASS architecture, that enables multi-scale collaboration and communication. In this paper we propose a time division multiplexing medium scheduling protocol tailored for this hierarchical data service, to take advantage of the communication and routing characteristics to achieve close to optimal latency and energy usage. We present an analytical proof on the bounds achieved by the protocol and analyze the performance via detailed simulations.

1 Introduction

Sensor networking has emerged as a promising tool for monitoring and actuating the devices of the physical world. It employs self-organizing networks of battery-powered wireless sensors that can sense, process, and communicate. Such networks can be rapidly deployed at low cost, enabling large-scale, on-demand monitoring and tracking over a wide area. The sensors can be deployed where it is difficult or resource-intensive to monitor the environment otherwise. Typical deployment examples are natural or man-made crises like severe weather, wild-fires, earthquakes, volcanic activities, chemical, biological or nuclear agents, and structural and habitat monitoring. In general, the sensors measure physical quantities at different spatial and temporal levels, supporting in-network signal processing and data assimilation. Queried results are next communicated with one

* This work was supported in part by NSF under grants CNS-0520280, CNS-0435425, CNS-0338856, and CNS-0325971; and by a gift from Schlumberger. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, Schlumberger, Aruba Networks, Rice University, or the U.S. Government or any of its agencies.

J. Aspnes et al. (Eds.): DCOSS 2007, LNCS 4549, pp. 386–403, 2007.

or more data sinks. Finally, based on the prevailing result, the sink nodes perform the desired computation to monitor and actuate the physical devices.

In this paper, we propose the design of an adaptive cross-layered scheduling service for a hierarchical network architecture providing for multi-scale collaboration. Energy and communication bandwidth are two of the most crucial and scarce resources in a sensor network. Also, sensor network applications generally exhibit application specific characteristics. Consequently, there is both a need and an opportunity for adapting the network architecture to match the applications. The objective is to minimize the resource consumption while extending the life of the network. There are specific requirements and limitations of sensor networks, which make their architecture and protocols both challenging and significantly different from that of traditional network architectures.

1.1 Motivating Application

The motivation to undertake this research came from the industrial need of sensor network architecture for real life applications. Deployment of sensor networks to monitor the ultra-clean environment for Intel fabrication plant [1] is a glaring example. The fabrication plants use some of the most delicate equipments to produce hundreds of thousands of chips daily. The fab process requires an ultra-clean environment with controlled temperature and humidity. This critical environment is maintained by a complicated cooling system composed of a diversity of pumps and coolers, amongst other machines, with multiple moving components. The breakdown of any of this equipment has a critical impact on the production line and results in significant losses. Today, Intel technicians manually carry around sensors to monitor abnormal vibrations in the equipment, which can detect worn bearings, failing compressors, and defective chillers. This vibration data is fed to an application which uses Fourier analysis to compare against expected profiles. When the analysis detects a variation in vibration outside the normal range, the affected equipment is scheduled for maintenance during normal down time.

Automation of this process makes it less dependent on human frailties—fatigue, oversight, and error—as well as enables more frequent monitoring for failures with rapid deterioration characteristics. Recently, Intel deployed a sensor network for gathering requisite operational data in an efficient way.

Real life deployment of proposed scheme demands robustness and optimization in many critical areas. Challenges in such a system include clustering together nodes of interest, providing communication paradigms for ease of application development, and efficient scheduling of communication between the battery-powered nodes. Optimizations possible include the following: (1) Temperature and vibration are monitored at different scales, and the reading of coarse granularity is reported frequently. Finer granularity readings of different scales are reported less frequently to conserve energy. For uni-dimensional data, like temperature or humidity, simple operations like computing the maximum or average suffice in the hierarchy of sensors. For multi-dimensional data, like vibration data, more complex compression algorithms can be utilized to reduce the data for communication. (2) In case of anomalous data, finer resolution sensor readings

are necessary to localize the area from which the anomaly was generated. Automatic alarms, with exact location and nature of the problem, are triggered for the operator, who then schedules immediate maintenance of the equipment.

1.2 COMPASS Overview

Many applications, such as large-scale collaborative sensing, distributed signal processing, and distributed data assimilation require the sensor data to be available at multiple resolutions, or allow fidelity to be traded-off for energy efficiency. The COMPASS architecture [2] we proposed enables scalability, localization and resolution-tuning, as well as provides communication abstractions to simplify application design. A few illustrations of applications that take advantage of this multi-scale approach include the following:

- *Multi-resolution Data Extraction*: A monitoring application requires variable resolution of sampled data. It requires higher resolution during operation at specific times of day or during periods of large variation, but lower resolution at other times. Also, a finite energy budget might be specified, and the resolution has to conform to this budget while providing the best possible resolution.
- *Requisite Resolution Drilling*: An application may demand relatively higher resolution data from a specific region. For example, the specific region might be a high security area thereby requiring closer monitoring, or the region might be deemed to be of greater interest because of the inherent nature of low-resolution data emanating from it. The specific region might actually have a fire, or may be generated due to malfunctioning sensors generating anomalous data.

The sensed data are the measured quantities governed by laws of physics. Consequently, there exists considerable temporal and spatial locality of measured data. For example, a sensor measuring temperature will have a high correlation with that of nearby sensors, as well as with its values sensed at earlier times. A simple hierarchical structure supporting multi-resolution will not be able to exploit such correlation effectively. A network hierarchy aligned to the communication flow can lead to operational efficiency in respect of volume and frequency of data to be transmitted. In this background we proposed a *self-organizing, self-adapting* sensor network architecture based on the specific requirements of an application.

The proposed architecture takes advantage of the fact that sensor network applications exhibit specific sets of behavior, compared to completely generalized network applications. The communication pattern in respect of source-destination pairs, as well as duration and periodicity, is known *a priori* in many applications. As opposed to multihop forwarding in ad hoc networks, the fusion of forwarded data in a sensor network opens the possibility of reduced communication, achieving higher efficiency of energy management. Hence, there exists opportunities to adapt the network protocols to meet the application requirements. Such adaptations require cross-layer optimizations in the networking stack, which goes beyond the strict protocol layering.

2 Adaptive Scheduling Service

This paper presents the design and evaluation of a scheduling protocol tailored for a multi-scale architecture. This energy efficient scheme enforces a tight bound on latency. It exploits the characteristics of sensor networks—periodic communication, limited communication abstractions, and known fusion function properties. To the best of our knowledge, scheduling based on information dependency of this type has not been attempted earlier. This scheduling takes into account the data flow as well as the aggregation performed at each hop. The scheduling protocol is adaptive: it is driven by the application demand and optimized for energy usage. Since no one single scheduling protocol is well-suited for all sensor applications [3], the network services need to adapt to the application-specific requirements. We design the scheduling protocol for data monitoring applications like that in the Intel fabrication plant noted earlier. Although the scheduling protocol has been optimized for such an application, it can support any generalized communication paradigm.

In the following sections, we present the background followed by a survey the related work. Next, we provide the design of the protocol, analyze its bounds, and report the simulation results.

2.1 Design Principles

In this section, we explore the design space for the scheduling protocol. The scheduling is a way to allow contending nodes to share a common channel by allocating non-conflicting time slots. The alternate to such a scheduling approach is contention-based access.

In shared-medium networks, one of the fundamental tasks of a medium access control (MAC) protocol is to avoid collisions between two interfering nodes. The protocol allocates the channel to the nodes efficiently, so that each node can communicate within a bounded waiting time and with as little overhead as possible. Some of the important attributes for traditional MAC protocols are fairness, latency, throughput, and bandwidth utilization. In contrast, the important attributes of a MAC protocol for WASN are energy efficiency and scalability towards size and topology change. The major sources of energy waste as elaborated by Ye et al. [4] are:

- **Collisions:** A collision results in corruption of a packet and subsequent retransmission, leading to increased energy consumption and latency.
- **Idle listening and overhearing:** Listening for packets addressed to this node or destined for other nodes wastes energy. Idle listening consumes significant energy comparable to actually receiving a packet.
- **Control packet overhead:** Increased control overhead consumes extra energy for transmitting and receiving these control packets.

The prior MAC protocols proposed [5,4] for WASN have identified and addressed many of the WASN *environment* requirements. However, the inherent advantages that can be derived from the specific characteristics of WASN have not been fully exploited. The following properties of WASN applications can be exploited for the MAC protocol design:

- Many sensor network applications have communication requirements that are *periodic* and known beforehand such as collecting temperature statistics at regular intervals. This periodic nature can be used to schedule the medium access by the nodes and thus minimize collisions. This can also aid the radio interfaces sleep/wake-up decisions and thereby decrease the idle listening and overhearing.
- A contention-based medium access scheme will also be necessary to support *event-driven* applications, such as intrusion or fire detection. The forwarding node can be woken up in time to process event-driven data by making it application-aware. Real-time constraints can be communicated from the application to adapt the MAC protocol to meet its deadlines.
- Frequently, sensor network applications are used for data gathering or monitoring, which implies that those communication flows are destined towards the sink. This single-destination communication characteristic can be taken advantage of in the MAC protocol to build efficient transmission schedules.
- In-network processing is used in sensor networks to reduce communication requirements. Knowledge of the aggregation characteristics can help determine a bound on the traffic requirements of each node.

The MAC protocol should have two modes to support the two different communication requirements of sensor applications, namely periodic and event-driven. The need to support these two kinds of modes was recently proposed in the IEEE standard for low-power sensors [6]. The relative proportion of the two modes in a frame can be dynamically determined by the applications, depending on the application's current needs.

Periodic Contention-Free Period: Medium access in this mode is of the class of Spatial Time Division Multiple Access (STDMA) [7] protocols. The application's deterministic traffic distribution during the periodic communication can be used to compute an efficient slot allocation policy. The length of each slot should be large enough to send a complete data packet of fixed a size.

Event-Driven Contention Access Period: During this mode, a sensor will be in sleep-mode except when necessary to communicate. The mode is based on IEEE 802.11 protocol [8] with carrier sense and RTS-CTS.

3 Related Work

There is a significant body of research that addresses the problem of providing access to the medium for next-hop communication. These protocols can be categorized as contention-based and schedule-based.

3.1 Contention-Based Protocols

IEEE 802.11 [8] is the best-known example of a contention-based MAC protocol. It uses a carrier sense multiple access technique combined with a handshake mechanism to access the channel and avoid collisions. A key limitation of IEEE 802.11 is that the nodes stay in idle mode for a long time, and hence this protocol is not suitable for

energy-constrained nodes. A simple Power Save Mode (PSM) is specified in order to reduce the energy consumption. Nodes are time-synchronized and awake at the beginning of a certain multiple of a beacon interval (called the Listening Interval and depends on the client configuration). It stays awake for a certain fixed period, and during this time, packets are exchanged to inform the nodes of buffered traffic. The nodes for which traffic is destined will remain awake after the fixed interval to receive the packets.

El-Hoiydi [9] proposed a low-level carrier sense technique that effectively duty cycles the radio. The basic idea is to shift the cost of data transfer from the receiver to the transmitter by increasing the length of the preamble. This leads to greater energy efficiency, as the number of receivers is more numerous than the number of transmitters in a broadcast network. In a recent implementation of this idea in the B-MAC procotol [10] as part of TinyOS, this preamble length is exposed as a parameter to the upper layers, so the application can select the optimal trade-off between energy savings and performance.

To reduce the overhead of periodic waking up and listening for incoming packets, there have been a number of proposals addressing this issue. The PAMAS protocol [11] was one of the first power-saving protocols that allows nodes to sleep when there is ongoing transmission in the neighborhood, leading to increased energy savings. S-MAC [4] adds a fixed slot structure and does duty cycling within each slot. At the beginning of each slot, nodes wake up and contend for the channel. The duty cycle can be directly controlled by the application for energy-performance trade-off, but the duty cycle must be fixed *a priori*. T-MAC [5] introduced an adaptive duty-cycle to automatically adjust to traffic fluctuations inherent in many applications.

All of the above techniques are useful for energy optimization when the traffic pattern is unknown. However, for situations when traffic pattern and routing path are known *a priori*, it makes sense to take advantage of this information for additional optimization.

3.2 Schedule-Based Protocols

Schedule-based MAC protocols are attractive because they are collision-free and there is no idle-listening. In this class of protocols, a slot is allocated for one node per neighborhood uniquely. This collision-free slot is used by that node for transmissions to any or all of it's neighbors. Thus two nodes cannot be assigned the same slot if one station is within the range of the other, or if two stations have common neighbors. The objective of these schedule-based protocols is to allow communication without interference, while maximizing the number of parallel transmissions.

Over the years, a substantial number of algorithms [12,7,13] have been published in an effort to solve the collision-free scheduling problem in multi-hop wireless networks. As the optimal scheduling is an NP-hard problem, all of these algorithms provide approximate solutions. These mostly centralized protocols are not optimized for sensor networks that have several unique characteristics compared to typical multi-hop ad hoc networks.

Bluetooth scheduling [14] is a closely related field of work. It enables formation of multi-hop networks using a combination of piconets, called scatternets. Each piconet has a master and up to seven slaves. Bluetooth does not use a global slot synchronization mechanism. Each link is individually synchronized by a reference provided by the

master in the link. Therefore, for multi-hop forwarding, slots are wasted as nodes need to switch time references. Various techniques have been proposed to address this time reference alignment problem. However, global synchronization is frequently required by the applications themselves in sensor networks, so it is reasonable to assume the existence of such a mechanism. Also, the master-slave structure in bluetooth is only one-hop and hence cannot be directly applicable to schedule the multi-hop trees formed by our proposed multi-scale architecture.

For multi-hop sensor networks, the TRAMA protocol [15] is one of the first proposals and is based on the NAMA protocol [16]. The nodes periodically awake to exchange broadcasts to learn the two-hop neighborhood. Using this knowledge, nodes reserve slots in the future for backlogged traffic. The protocol uses a hash function for detecting priority among contending neighbors. This leads to priority chaining whereby nodes can get higher priority in one neighborhood and lower in another, causing inefficient scheduling. TRAMA tries to schedule packets hop-by-hop instead of flows, leading to higher end-to-end latency.

Sichitiu's work [17] is the closest to ours with regard to the fact that it tries to schedule flows end-to-end. It dynamically sends a *SETUP* packet to schedule new flows. If this packet is received without collision by the receiver, this slot is scheduled for this pair of neighbors. This receiver then uses the same mechanism for forwarding the packet over the next hop towards the data sink, thereby creating a scheduled flow for the whole path. This approach is similar to the one we use for scheduling flows in our inter-cluster scheduling. However, our intra-cluster protocol takes advantage of the fact that sensed data are processed at each hop in many typical sensor network deployments, leading to better compression and lesser number of slots necessary. In Sichitiu's protocol, each flow is individually scheduled, leading to higher latency in data sensing applications.

3.3 Graph Coloring

The scheduling problem can be directly mapped to the graph coloring problem. Minimum graph coloring is a well-known NP-hard problem and has been researched for several decades in order to find solutions closest to the optimal. In the graph, a valid distance-2 coloring assigns different colors to any pair of nodes between which there is a path length of at most 2; this is very similar to producing a conflict-free schedule. A good background work based on this is provided by Marathe et al. [18]. However, graph coloring corresponds only to the 2-hop conflict-free property necessary for scheduling. In our scheduling protocol, we have additional constraints on slot allocation due to which previous graph coloring techniques cannot be directly applied. These constraints are elaborated in the next section.

4 Medium Access Scheduling in a Hierarchy

This section describes the design of the medium access protocol for a hierarchically clustered network. This medium access protocol is specifically designed for supporting the communication abstractions supported in this architecture. There is a Conflict Free Period (CFP) for periodic *a priori* known traffic, and a Contention-based Access Period

(CAP) for nodes to send event-driven traffic, and to introduce new periodic traffic. A TDMA-style protocol is designed for the CFP, and we leverage the existing literature on energy-efficient CSMA protocols [19,4,5] for the CAP.

A time frame in the protocol is shown in Figure 1. Interval L_1 interval corresponds to communication amongst nodes within the first-level cluster. A near optimal protocol proposed to schedule intra-cluster communication is the main contribution in this chapter, and is elaborated in Section 4.1. Interval L_i for $i > 1$, is the time interval during which communication takes place between cluster-heads at level i. The intervals for communicating among different levels are separated so as minimize the perturbation at higher levels when localized change takes place in lower-level schedules. The total length of this CFP period is kept as small as possible for two reasons: (1) the CFP period gives the lower bound on the real-time responsiveness of the system; and (2) the nodes need to stay awake only during part the CFP period.

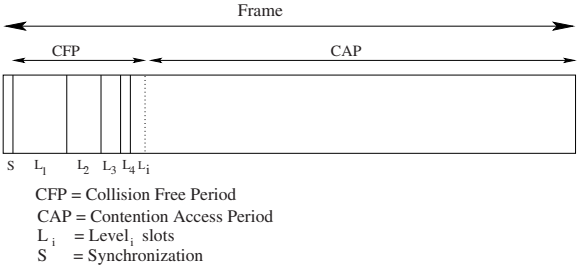

Fig. 1. Frame diagram

We make the following assumptions in the design of the protocol:

- Wireless links used for sending or receiving packets are bi-directional. This assumption is typical for most MAC protocols.
- The interference range is the same as the range for transmission and reception. The interference range depends on a variety of factors and cannot be represented easily as a function of the transmission range. Most collision-free scheduling work on the approximation of equality between transmission and interference range.
  This assumption can be relaxed using techniques such as RID [20], where instead of 2-hop communication to reach interfering nodes, multi-hop forwarding is used to reach the interfering nodes. My theory of conflict-free slot allocation proposed in this design requires the ability to ascertain and communicate with interfering nodes. Thus, our protocol can be adapted to use RID in future, as RID provides these.
- A clock synchronization facility [21] exists.

This new scheduling protocol is designed specifically for multi-scale data retrieval applications, where data flows from the sensors to the sink. This is the common case usually encountered in a typical sensor network application.

Algorithm 1. Centralized Algorithm for Intra-Cluster Scheduling

Require: k slots necessary per node known; Depends on fusion function and number of children

1: Traverse tree in Depth First order from cluster root
2: Finish order of the visit forms a topological sort to give a partial order
3: For each visited node, assign a minimum possible slot number s, with the following constraints
4: **repeat**
5: $s > \max$ (slot number allocated to any child)
6: $s \neq$ slot already allocated to any node within 2 hops
7: **until** k consecutive slots are found

4.1 Intra-cluster Scheduling

Protocol. The chief design goal of this scheduling protocol is to allocate spatially conflict-free slots for each node within a cluster. A tree-like hierarchy within each lowest-level cluster has already been created by the routing protocol. The other design goals of this protocol are as follows:

1. **Partial order:** As the data flows upwards from the children to their parent, the parents are allocated slots subsequent to the allocated slots of their children. This property allows data from the lowest level sensor to flow to the top-level sink within one frame, thereby decreasing the latency for data gathering per epoch. This induces a partial ordering of the nodes to be maintained by the scheduling scheme.
2. **Contiguous allocation:** The slots allocated to the children of a node should be close together, such that the parent can wake up for one contiguous time to receive data packets from its children, and can then switch off the radio to save power. As switching between on and off state consumes time and energy, contiguous allocation is beneficial. This entails a scheduling policy in which the slots of siblings needs to be grouped together.
3. **Fusion characteristics:** The fusion function characteristics at each node are known *a priori*, such as through the Information Exchange Service (IES) [22]. Fusion function characteristics are analyzed by the scheduling protocol to determine the number of slots to be allocated at each level. For example, for a fusion function such as average sensed value, the parent needs to average its own value with the value received from each of its children and then send it upwards in a single slot. The number of slots to be allocated to a node per frame also depends on the application-determined periodic rate, which can also be known through the IES. Therefore, a node might have multiple slots allocated, as deemed necessary by the fusion function.

We first describe a base centralized scheduling scheme, that allocates conflict-free slots and later generalize it to a distributed scheduling scheme. The centralized scheme is shown in Algorithm 1, Within the lowest level cluster, a cluster-head coordinates the traffic out of that cluster. It does a Depth-First Search (DFS) of the tree rooted at itself, and sorts the nodes according to the finish order (of a DFS visit) to give a

post-order traversal of the nodes within the cluster. The post-order traversal has the desirable property that the parent has a finish order immediately after the finish order of its children in the tree. The slot times are assigned according to that given by the topological sort, thereby providing the desired partial order. This algorithm produces a collision-free slot allocation within a single cluster. However, this centralized scheme has the following three problems: (1) It assumes a complete graph knowledge. (2) It leads to a very inefficient slot allocation, with no concurrent communication at all. This produces a large CFP, which is detrimental for real-time response. This is shown in Figure 2, where 7 slots are used for 7 nodes. In the figure, the lines show the parent-child relationship in the cluster and the dotted lines show that that nodes at the two ends of that line are within range of each other. (3) Inter-cluster interference is not accounted for.

We next describe a distributed token-passing protocol to compact the slot allocation and meet all the other design goals. This algorithm is formally described in Algorithm 3. The algorithm requires each node to know its 2-hop neighborhood, the cluster it belongs to, and its parent and children. The cluster root generates a token packet and passes the token around in DFS order. The node holding the token allocates a slot for itself when the token comes back to it after traversing all its children. The allocated slot is put inside the token, and hence the token contains the allocated slots of all the nodes it has traversed. As the siblings need to be allocated slots as close to each other as possible, a slot is chosen according to this condition. There is a trade-off possible between choosing closer slots to minimize the number of transitions, or choosing the minimum possible slot to minimize the total number of slots needed for CFP.

Choosing the next child to visit in the DFS can be random (which we call the Base algorithm) or based on some graph property. Based on observation from Figure 4 and experimental comparison with a few other techniques, we order the children based on their degree and choose the next child to visit based on that order. The other techniques we compared with are: ordering based on the number of children each node has, and ordering based on the number of nodes in the subtree rooted at the node. The intuitive reason is to allocate the lesser slots to the nodes in a dense area, thereby leading to less conflict allocating slots in the less dense areas. We call this version of the algorithm the Degree algorithm.

Analysis. This problem can be mapped to a graph coloring problem as follows. The network is taken as a graph (G), and a graph G^2 is produced. G^2 has the same vertex set as G, and all the pairs of vertices which are 2-edge reachable in G have an edge in G^2. Thus, G^2 gives the interference graph similar to our node scheduling problem. Coloring a graph is an NP-hard problem even in a centralized setting, and many approximate coloring solutions have been proposed in the graph theory literature. However, due to the constraints on slot allocation induced by the partial ordering, none of these solutions can be directly applied for our scheduling scenario. The graph being that of a wireless network, it is an Unit Disk Graph (UDG) having special properties. In a UDG, a link between two nodes exists if the distance between the nodes is less than the radio range. The *competitive ratio* is the ratio between the chromaticity given by the approximate algorithm and the optimal solution.

Algorithm 2. Distributed Protocol for Intra-Cluster Scheduling

Require: k slots necessary per node known; Depends on fusion function and number of children
Require: Each node knows its 2-hop neighborhood
Require: Each node knows the cluster to which it belongs
Require: Each node knows its parent and its children
For all token packets overheard, a node remembers slots its neighbor is sending or receiving
while All children of a node not visited **do**
    Send token to one of the children not visited. The child to visit next next depends on the heuristic chosen.
    Wait until token comes back
    Then remember the slot allocation of the child
end while
All nodes in the subtree rooted at the node has been visited
For each visited node, assign a possible slot number s, with the following constraints
repeat
        $s >$ maximum (slot number allocated to any child)
        $s \neq$ slot already allocated in token packet
        $s \neq$ slot overheard being allocated by another node in 1-hop neighborhood
        Assign the k slots close to slots already allocated by it's siblings
until k consecutive slots are found
Append the node id, its chosen sending slot, and its send mode (parent or all) to the token
Append the node id and receiving slots to the token
Send token to its parent with flag stating SUBTREE_DONE
while Token not received from parent with flag SUBTREE_DONE **do**
    Wait
end while
Broadcast the final token packet locally

Lower bound on optimality: We show using a sub-graph, which we believe is the worst-case for our algorithm, that the bound of this competitive ratio is 1.2 for our Base algorithm. Figure 3 and Figure 4 show the nodes in a cluster labled A through H. They form a tree hierarchy denoted via the lines. The dotted lines show two nodes within range of each other. The numbers beside the nodes shows the slot that the node has been allocated. Comparing the worst case slot allocation in Figure 3 with the optimal slot allocation for the same graph in Figure 4, we get the 1.2 ratio. So, if the adversary chooses the graph as well as the ordering of DFS visits in the base algorithm, our protocol cannot expect an optimality lesser than that value.

Upper bound on optimality: Employing the properties of an UDG, an upper bound of 6 has been also established for the competitive ratio for the base algorithm. This is based on the observation that there can be at most 5 neighbors of a node which are not connected to each other. If in a naive approach our algorithm gives different color to all of its neighbors, it can at most be 6 times as bad, as there will be a clique in the size of degree 6.

This small constant bound on the competitive ratio is a good indication of the optimality of our proposed solution. The time and cost for the scheduling protocol are both $O(N)$ where N is the number of nodes in the smallest level cluster. The clustering

Algorithm 3. Distributed Protocol for Intra-Cluster Scheduling

Require: k slots necessary per node known; Depends on fusion function and number of children
Require: Each node knows its 2-hop neighborhood
Require: Each node knows the cluster to which it belongs
Require: Each node knows its parent and its children
For all token packets overheard, a node remembers slots its neighbor is sending or receiving
while All children of a node not visited **do**
    Send token to one of the children not visited. The child to visit next next depends on the heuristic chosen.
    Wait until token comes back
    Then remember the slot allocation of the child
end while
All nodes in the subtree rooted at the node has been visited
For each visited node, assign a possible slot number s, with the following constraints
repeat
        $s >$ maximum (slot number allocated to any child)
        $s \neq$ slot already allocated in token packet
        $s \neq$ slot overheard being allocated by another node in 1-hop neighborhood
        Assign the k slots close to slots already allocated by it's siblings
until k consecutive slots are found
Append the node id, its chosen sending slot, and its send mode (parent or all) to the token
Append the node id and receiving slots to the token
Send token to its parent with flag stating SUBTREE_DONE
while Token not received from parent with flag SUBTREE_DONE **do**
    Wait
end while
Broadcast the final token packet locally

hierarchy ensures that the number of nodes in a cluster is constant, subject to the constant node density. So, the algorithm is both constant in time and cost with the number of nodes in the network.

We have also proved the upper and lower bounds on the number of slots used in the graph.

Theorem 1. The upper bound on the number of slots used is $\Delta^2 + h$, where Δ is the maximum node degree and h is the height of the tree.

Proof. Δ^2 implies the 2-hop neighborhood of a node. As We allocate the nodes in conflict-free 2-hop neighborhood, the whole neighborhood can always be colored using the same number of colors as there are nodes. Hence, the whole graph can be colored using the number of colors in the maximum degree of any node. For uniform random distribution of nodes, we take Δ as the average degree.

The h factor comes in as one slot increases for each level of the tree. The parent needs to be strictly one slot more than its children.

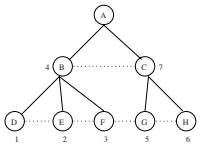

Fig. 2. Centralized slot allocation in a subgraph

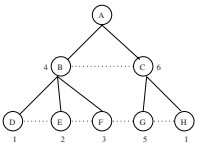

Fig. 3. Possible slot allocation in a worst-case subgraph

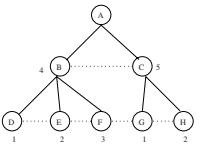

Fig. 4. Optimal slot allocation in a worst-case subgraph

Theorem 2. The lower bound on the number of slots used is $\Delta + h$, where Δ is the maximum node degree and h is the height of the tree.

Proof. As a 2-hop neighborhood is to be made conflict-free of every node, the range of connectivity of nodes becomes $2 \times r$, where r is the range of the node. So, within a circle of r radius, all nodes are within 2-hops of each other and hence form a clique—a fully connected graph. So, atleast Δ colors are needed to color this clique.

As before, the h factor comes in as one slot increases for each level of the tree. The parent needs to be strictly one slot more than its children.

4.2 Inter-cluster Scheduling

Inter-cluster scheduling occurs after the data has reached the cluster-heads at the lowest level cluster. Each level of communication is separated into time, and hence does not conflict with each other. This is shown in Figure 1, where the CFP is divided into multiple periods for each level.

For inter-cluster communication between peers, the medium access protocol utilizes knowledge of the routing path. Beacons transmitted by cluster-heads at all levels of the hierarchy by the routing protocol specifies the routing path for communication between peers at different levels. A packet reserving the channel is sent for the periodic traffic along the routing path specified by the beacons. Each node in the path allocates slots in a conflict-free fashion such that two different peer-paths do not conflict in time, while ensuring maximum parallelism in the communication paths.

5 Evaluation

To evaluate the performance of this proposed protocol, we measure the latency of data gathering and associated energy savings. As we utilize extra cross-layer information, which no previous approach has used before, we compare against the optimal solution.

We used a simulator to randomly generate graphs, construct a tree in the way described in the routing protocol, and then implement the scheduling protocol. The simulator helped in the validation and experimentation with a number of different heuristics to select the best one. Each of the points was generated as an average of 50 runs.

Finding the optimal solution for scheduling was the primary reason we used a graph simulator, as using a traditional simulator with full simulation of physical and medium

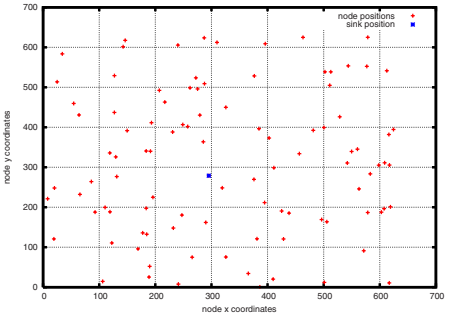

Fig. 5. Uniform random distribution of 100 nodes with average degree of 8

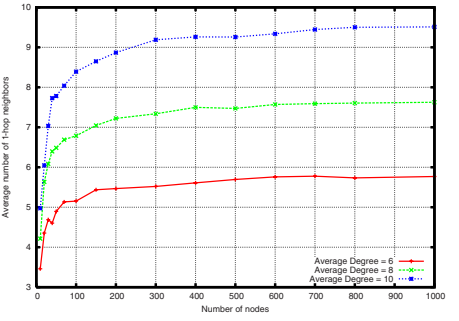

Fig. 6. Number of 1-hop neighbors with number of nodes, for different degree

access later would have been too slow to generate the optimal solution. We implemented simulated annealing technique to get the optimal solution needing exponential complexity. Up to 10 million runs were necessary for finding each point on the optimal graphs.

The data gathering latency is directly related to the total number of slots needed to cover the entire network $\times$ the duration of each slot. Hence, we show the number of total slots as a measure of latency. The energy savings come from the amount of time (or slots) each node has to remain in active mode to receive or transmit data or idle. This is also calculated as the average number of slots each node has to remain awake.

Figure 5 shows a random uniform distribution of 100 nodes with an average degree of 8, and range of 100. This corresponds to a square with side as 626.6. The node closest to the center is chosen as the sink node. All graphs have the number of nodes varying from 10 to 1000, and average degree of 6, 8 and 10.

As the analytical upper and lower bounds depend on the 1-hop and 2-hop neighbors, the neighborhood degrees are plotted with increasing number of nodes. Figure 6 shows the plot for degree of 6, 8 and 10. The plots approach those values with increasing number of nodes, but are lesser because of edge effects which lower the average.

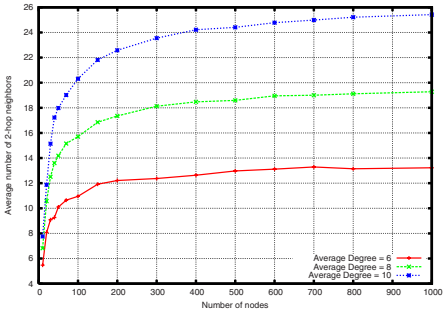

Fig. 7. Number of 2-hop neighbors with number of nodes, for different degree

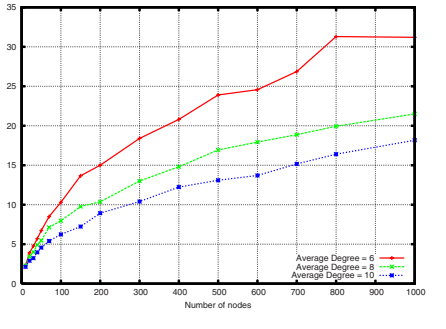

Fig. 8. Height of tree with number of nodes, for different degree

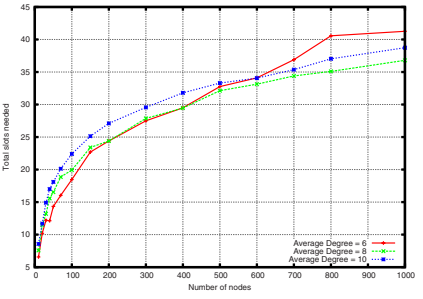

Fig. 9. Latency with number of nodes, for different degree

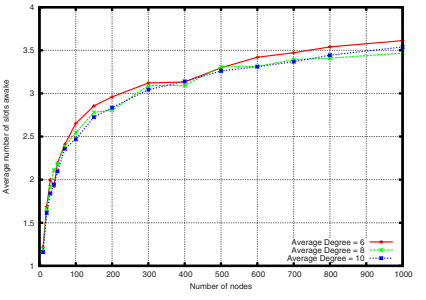

Fig. 10. Average number of slots in which each node is awake, for different degree

Figure 7 is similarly a plot of the number of 2-hop neighbors for a degree of 6, 8 and 10. The number of nodes is proportional to the area for a given uniform density. If one takes the ratio of the area of two circles, with twice the radius in one from the other, it will have a value of 4. But, in the graph the value is about 2.5 times the 1-hop neighbor value. Other than edge effects, another factor which reduces it is the density of nodes. The ratio of 4 will be reached only in a very dense topology where a 2-hop route covers all the area of a circle with twice the radius.

The analytical bounds also depends on the depth of the tree formed. Hence, the depth of the tree formed for increasing number of nodes is shown in Figure 8. The height increases logarithmically as has been analytically shown before. The curve for degree 6 leads to poor connectivity and frequently forms disconnected graph, specially at larger number of nodes. It also leads to large depths due to long circuitous paths because of lack of density. That is the reason for the depth curve not being smooth like the curves for other degrees. For 500 nodes, average degree of 10 leads to height of about 12. Figure 9 shows the latency for gathering data from every sensor up to the sink node. It is given in terms of slots, where each slot takes the duration equivalent to the transmission of the largest data packet. The number of slots necessary increases logarithmically with n, thereby scaling to a large number. For 500 nodes in a cluster, about 33 slots are needed for degree equal to 10.

The average number of slots each node needs to be awake for is presented next in Figure 10. For 500 nodes, this value is about 3.2. The number of slots idle or transmitting or receiving is inversely proportional to the lifetime of the node. The average number of slots awake increases very slowly with increasing number of nodes, also pointing to the scalability of the system. Assuming 500 byte payload and 100 Kbps radio, approximate slot size would be 5 ms. So, on an average each node needs to be awake for $5 \times 3.2 = 16$ ms. So, if data monitoring frequency is 1 event per second, a node has to be awake only 1.6% of the time.

Figure 11 shows the analytical bounds proved earlier, as well as result from running experiments to find the optimal solution. The upper bound as well as lower bound is strictly met by the optimal graph.

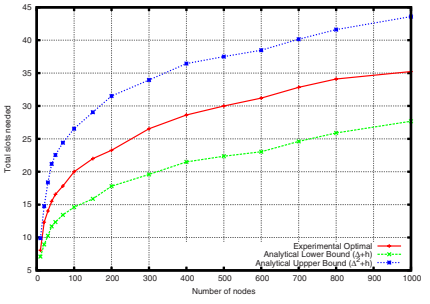

Fig. 11. Analytical bounds on optimality

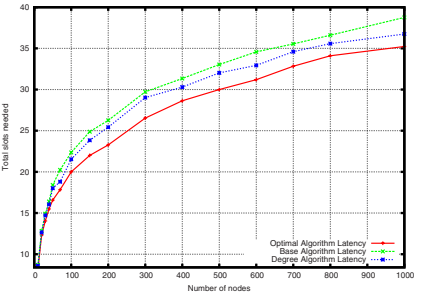

Fig. 12. Comparison of the latency for different algorithms

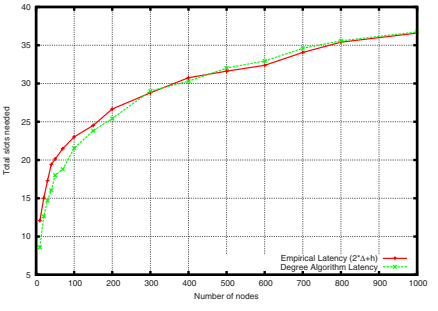

Fig. 13. Empirical and experimental latency

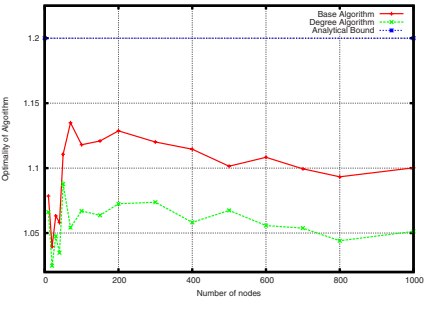

Fig. 14. Normalized optimality of algorithm

Figure 12 compares the latency of the base and the degree algorithm with that of the optimal. Normalized optimality if shown in Figure 14. The optimality reduces and stabilizes at 1.05 for Degree algorithm, and 1.1 for Base Algorithm. We analytically found a bound of 1.2 for the base algorithm on a worst-case graph. A value of $2\times\Delta+h$ is empirically plotted and it is seen that it closely follows the experimental optimal for the degree algorithm. This total number of slots gives the chromatic number of the graph, and there does not exist any polynomial solution to this. However, this empirically observed formula seems to match closely with the experimental results.

These results show that the scheduling protocol is very close to optimal scheduling in terms of total latency of data gathering, as well as in terms of energy efficiency.

6 Conclusions and Future Work

Apart from enabling multi-resolution collaboration, a clustering hierarchy allows the network to scale to a very large number of sensors. Our architecture design adapts to the communication and collaboration requirements of the application, reducing communication energy and bandwidth usage. This work highlights the need for and possibility

of adapting a scheduling protocol to suit the application requirements. By making the protocol adaptable to the application needs dynamically, the new protocol will allow exploiting the application-specific requirements.

We have not addressed sensor network reliability or QoS requirements. These are important aspects that deserve attention. Abstractions with tunable parameters through which the application can control the trade-off between resource usage and accuracy and reliability can be developed. Abstract Regions currently provides a tuning interface, but it requires the application to specify low-level parameters such as number of retransmissions. The tunable parameter needs to be at a higher level, which will enable the application to set goals, that will be automatically translated by the networking layer into the low-level parameters.

Making the routing and scheduling adapt without application aid can be another future direction of work. Instead of explicit communication from the application as is done in this paper, information flow can be learned. Thereafter, the adaption is done based on this learning.

References

1. Chhabra, J., Kushalnagar, N., Metzler, B., Sampson, A.: Sensor networks in intel fabrication plants. In: SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM Press, New York (2004)
2. PalChaudhuri, S., Kumar, R., Baraniuk, R., Johnson, D.: Design of adaptive overlays for multi-scale communication in sensor networks. In: Prasanna, V.K., Iyengar, S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, Springer, Heidelberg (2005)
3. Heidemann, J., Silva, F., Estrin, D.: Matching data dissemination algorithms to application requirements. In: SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 218–229. ACM Press, New York (2003)
4. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient mac protocol for wireless sensor networks. In: Proceedings of the IEEE Infocom, New York, NY, USA, USC/Information Sciences Institute, IEEE, pp. 1567–1576 (June 2002)
5. van Dam, T., Langendoen, K.: An adaptive energy-efficient mac protocol for wireless sensor networks. In: Proceedings of the first international conference on Embedded networked sensor systems, pp. 171–180. ACM Press, New York (2003)
6. IEEE Computer Society LAN MAN Standards Committee: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4. The Institute of Electrical and Electronics Engineers, New York (2003)
7. Nelson, R., Kleinrock, L.: Spatial-TDMA: A collision-free multihop channel access control. IEEE Transactions on Computers 33, 934–944 (1985)
8. IEEE Computer Society LAN MAN Standards Committee: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York (1997)
9. El-Hoiydi, A.: Aloha with Preamble Sampling for Sporadic Traffic in Ad-hoc Wireless Sensor. In: Proceedings of IEEE International Conference on Communications (ICC), New York, USA (2002)
10. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 95–107. ACM Press, New York (2004)

11. Singh, S., Raghavendra, C.: PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. SIGCOMM Computer Communication Review, vol. 28(3) (July 1998)
12. Chou, A.M., Li, V.: Slot allocation strategies for TDMA protocols in multihop packet radio networks. In: Proceedings of INFOCOM 1992, pp. 710–716 (1992)
13. Chlamtac, I., Farago, A.: Making transmission schedules immune to topology changes in multi-hop packet radio networks. IEEE/ACM Trans. Netw. 2(1), 23–29 (1994)
14. Salonidis, T., Tassiulas, L.: Asynchronous TDMA ad hoc networks: Scheduling and Performance. In: Proceedings of European Transactions in Telecommunications (ETT) (2004)
15. Rajendran, V., Obraczka, K., Garcia-Luna-Aceves, J.J.: Energy-efficient collision-free medium access control for wireless sensor networks. In: Sensys '03: Proceedings of the first international conference on Embedded networked sensor systems, pp. 181–192. ACM Press, New York (2003)
16. Bao, L., Garcia-Luna-Aceves, J.J.: A new approach to channel access scheduling for ad hoc networks. In: MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 210–221. ACM Press, New York (2001)
17. Sichitiu, M.: Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. In: Proceedings of INFOCOM 2004 (2004)
18. Marathe, M., Breu, H., Ravi, H., Rosenkrantz, D.: Simple heuristics for unit disk graphs. Networks 25, 59–68 (1995)
19. Woo, A., Culler, D.E.: A transmission control scheme for media access in sensor networks. In Mobile Computing and Networking, pp. 221–235 (2001)
20. Zhou, G., He, T., Stankovic, J.A., Abdelzaher, T.F.: Rid: Radio interference detection in wireless sensor networks. In: Proceedings of IEEE Infocom (2005)
21. PalChaudhuri, S., Saha, A., Johnson, D.B.: Adaptive clock synchronization in sensor networks. In: Proceeding of the Information Processing in Sensor Networks(IPSN), Berkeley, CA (April 2004)
22. Kumar, R., PalChaudhuri, S., Ramachandran, U.: System support for cross-layering in sensor network stack. In: Proceedings of the International Conference on Mobile Ad Hoc and Sensor Networks, Hong Kong, China (December 2006)